

# Linear-memory and Decomposition-invariant Linearly Convergent Conditional Gradient Algorithm for Structured Polytopes

Dan Garber and Ofer Meshi  
Toyota Technological Institute at Chicago  
{dgarber, meshi}@ttic.edu

## Abstract

Recently, several works have shown that natural modifications of the classical conditional gradient method (aka Frank-Wolfe algorithm) for constrained convex optimization, provably converge with a linear rate when: i) the feasible set is a polytope, and ii) the objective is smooth and strongly-convex. However, all of these results suffer from two significant shortcomings:

1. large memory requirement due to the need to store an explicit convex decomposition of the current iterate, and as a consequence, large running-time overhead per iteration
2. the worst case convergence rate depends unfavorably on the dimension

In this work we present a new conditional gradient variant and a corresponding analysis that improves on both of the above shortcomings. In particular:

1. both memory and computation overheads are only linear in the dimension
2. in case the optimal solution is sparse, the new convergence rate replaces a factor which is at least linear in the dimension in previous works, with a linear dependence on the number of non-zeros in the optimal solution

At the heart of our method, and corresponding analysis, is a novel way to compute decomposition-invariant *away-steps*. While our theoretical guarantees do not apply to any polytope, they apply to several important structured polytopes that capture central concepts such as paths in graphs, perfect matchings in bipartite graphs, marginal distributions that arise in structured prediction tasks, and more. Our theoretical findings are complemented by empirical evidence which shows that our method delivers state-of-the-art performance.

## 1 Introduction

The efficient reduction of a constrained convex optimization problem to a constrained linear optimization problem is an appealing algorithmic concept, in particular for large-scale problems. The reason is that for many feasible sets of interest, the problem of minimizing a linear function over the set admits much more efficient methods than its non-linear convex counterpart. Prime examples for this phenomenon include various structured polytopes that arise in combinatorial optimization, such as the path polytope of a graph (aka the unit flow polytope), the perfect matching polytope of a bipartite graph, and the base polyhedron of a matroid, for which we have highly efficient combinatorial algorithms for linear minimization that rely heavily on the specific rich structure of the polytope [22]. At the same time, minimizing a non-linear convex function over these sets usually requires the use of generic interior point solvers that are oblivious to the specific combinatorial structure of the underlying set, and as a result, are often

much less efficient. Another important example includes structured sets of matrices such as the *spectrahedron*, i.e., convex-hull of unit-trace positive semidefinite matrices, or the *nuclear ball* that are central to many machine learning problems, such as *matrix completion*, for which linear optimization amounts to computing the leading eigenvector or leading pair of singular vectors, whereas, algorithms for non-linear convex optimization over these sets often rely on very expensive singular value decompositions. Indeed, it is for this reason, that the conditional gradient (CG) method (aka Frank-Wolfe algorithm), a method for constrained convex optimization that is based on solving linear subproblems over the feasible domain, has regained much interest in recent years in the machine learning, signal processing and optimization communities. It has been recently shown that the method delivers state-of-the-art performance on many problems of interest, see for instance [13, 17, 4, 9, 10, 23, 19, 26, 11, 14].

As part of the regained interest in the conditional gradient method, there is also a recent effort to understand the convergence rates and associated complexities of conditional gradient-based methods, which is in general far less understood than other first-order methods, e.g., the projected gradient method. It is known, already from the first introduction of the method by Frank and Wolfe in the 1950's [5], and the somewhat later work of Polyak and Levitin [20], that the method converges with a rate of roughly  $O(1/t)$  for minimizing a smooth convex function over a convex and compact set, which matches the rate of the standard projected gradient method for the same setting. However, it is not clear if this convergence rate improves under an additional standard strong-convexity assumption. In fact, certain lower bounds, such as in [18, 6], suggest that such improvement, even if possible, should come with a worse dependence on the problem's parameters (e.g., the dimension), which is a phenomena that does not occur for the projected gradient method, for instance. Nevertheless, over the past years, various works tried to design natural variants of the CG method that converge provably faster under the additional strong convexity assumption, or a slightly weaker assumption, without dramatically increasing the per-iteration complexity, which is the main appeal for these methods. For instance, GuéLat and Marcotte [8] showed that a CG variant which uses the concept of *away-steps* converges exponentially fast in case the objective function is strongly convex, the feasible set is a polytope, and the optimal solution is located in the interior of the set. A similar result was presented by Beck and Teboulle [3] who considered a specific problem they refer to a *the convex feasibility problem* over an arbitrary convex set. They also obtained a linear convergence rate under the assumption that an optimal solution that is far enough from the boundary of the set exists. In both of these works, the exponent depends on the distance of the optimal solution from the boundary of the set, which in general can be arbitrarily small. Later, Ahipasaoglu, Sun and Todd [1] showed that in the specific case of minimizing a smooth and strongly convex function over the unit simplex, a variant of the CG method which also uses away-steps, converges with a linear rate. Unfortunately, it is not clear from their analysis how this rate depends on natural parameters of the problem such as the dimension and the condition number of the objective function.

Recently, Garber and Hazan presented the first natural linearly-converging CG variant for polytopes without any restricting assumptions on the location of the optimum. The exponent in their convergence rate depends on various geometric parameters of the polytope [6]. It is important to note, that while, in theory, these geometric parameters can result in an arbitrarily bad convergence rate, for polytopes for which it makes sense to apply the CG method, i.e., there exists an highly efficient algorithm to solve the linear subproblems, such as polytopes that arise in combinatorial optimization problems, these parameters are quite reasonable and can be efficiently computed. In a follow-up work, Lacoste-Julien and Jaggi [15, 16] gave a refined affine-invariant analysis of an algorithm presented in [8] which also uses away steps, and showed that it also converges exponentially fast in the same setting as the Garber-Hazan

result. In a later work, Beck and Shtern [2] gave a different, duality-based, analysis for the algorithm of [8], and showed that it can be applied to a wider class of functions than purely strongly convex functions. However, the explicit dependency of their convergence rate on the dimension is suboptimal, compared to [6, 16]. Aside from the polytope case, Garber and Hazan have shown recently that in case the feasible set is strongly-convex and the objective function satisfies certain strong convexity-like properties, then the standard CG method converges with an accelerated rate of  $O(1/t^2)$  [7].

Despite the exponential improvement in convergence rate in the polytope case obtained in recent results, all of these results suffer from two major drawbacks. First, while in terms of the number of calls per-iteration to the linear optimization oracle, these methods match the standard CG method, i.e., a single call per iteration, the overhead of other operations both in terms of running times and memory requirements is significantly worse. The reason is that in order to apply the so-called away-steps, which all methods use in order to obtain the accelerated rate, they require to maintain at all times an explicit decomposition of the current iterate into vertices of the polytope. Maintaining such a decomposition and computing the away-steps, even with efficient implementations of incremental decomposition procedures, such as suggested in [2], require both memory and per-iteration runtime overheads that are at least quadratic in the dimension. This is much worse than the standard CG method, whose memory and runtime overheads are only linear in the dimension. Second, the convergence rate of all previous linearly convergent CG methods depends explicitly on the dimension. While it is known that this dependency is unavoidable in certain cases, e.g., when the optimal solution is, informally speaking, dense (see for instance the lower bound in [6]), it is not clear that such an unfavorable dependence is mandatory when the optimum is sparse.

In this paper, we revisit the application of CG variants to smooth and strongly-convex optimization over polytopes. We introduce a new variant which overcomes both of the above shortcomings from which all previous linearly-converging variants suffer. The main novelty of our method, which is the key to its improved performance, is that unlike previous variants, it is decomposition-invariant, i.e., it does not require to maintain an explicit convex decomposition of the current iterate. This principle proves to be crucial both for eliminating the memory and runtime overheads, as well as to obtaining sharper convergence rates for instances that admit a sparse optimal solution.

We give a detailed comparison of our method to previous art in Table 1. We also provide empirical evidence that the proposed method delivers state-of-the-art performance on several tasks of interest. While our method is less general than previous ones, i.e., our theoretical guarantees do not hold for arbitrary polytopes, they readily apply for many structured polytopes that capture important concepts such as paths in graphs, perfect matchings in bipartite graphs, Markov random fields, and more. We also specify how to apply the method to arbitrary polytopes, but without giving formal convergence guarantees.

## 1.1 Organization of the paper

The rest of this paper is organized as follows. In Section 2 we give preliminaries and notation, and present the exact setting considered in this paper. In Section 3 we briefly present the conditional gradient method and its previous away-steps-based variants, and present our new method: a decomposition-invariant pairwise conditional gradient algorithm. In this section we also give our main theorem which details the novel convergence rate of our method. In Section 4 we briefly describe several important polytopes that fall into our assumptions, and detail the application of our method for optimization over these polytopes. In Section 5 we give a complete analysis of our method and prove the main theorem. In Section 6 we detail how to apply our

Paper	#iterations to $\epsilon$ err.	#LOO calls	runtime	memory
Frank & Wolfe [5]	$\frac{\beta D^2}{\epsilon}$	1	$n$	$n$
Garber & Hazan [6]	$\frac{n\beta D^2}{\alpha} \log(1/\epsilon)$	1	$n^2$	$n^2$
Lacoste-Julien & Jaggi [16]	$\frac{n\beta D^2}{\alpha} \log(1/\epsilon)$	1	$n^2$	$n^2$
Beck & Shtern [2]	$\frac{n^2\beta D^2}{\alpha} \log(1/\epsilon)$	1	$n^2$	$n^2$
This paper	$\frac{\text{card}(x^*)\beta D^2}{\alpha} \log(1/\epsilon)$	2	$n$	$n$

Table 1: Comparison with previous works. The third column gives the number of calls to the linear optimization oracle per iteration, fourth column gives the overall additional arithmetic complexity per iteration, and the fifth column gives the worst case memory requirement of the algorithm. To get lower complexity and memory requirements for the algorithms in [6, 16, 2], we assume they all employ an algorithmic version of Carathodory’s theorem to maintain a convex decomposition of the iterate to at most  $n + 1$  vertices, as fully detailed in [2]. We note that the bound on number of iterations in the analysis of [16] does not depend explicitly on the dimension  $n$ , but on the squared inverse *pyramidal width* of  $\mathcal{P}$ , which is difficult to evaluate. However, already for the simplest polytope, i.e., the unit simplex, this quantity is proportional to  $n$ .

approach to a broader class of polytopes, though we do not complement our algorithm with a convergence rate result in this case. We also show that our requirement that the objective function is strongly convex can be relaxed, and that our results in fact hold for a broader class of functions. In Section 7 we introduce a lower-bound for conditional gradient-based methods, that shows that for certain problems with a sparse optimal solution, our method is nearly optimal. Finally, in Section 8 we present empirical evidence which demonstrates the performance of our method.

## 2 Preliminaries

**Definition 1.** We say that a function  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $\alpha$ -strongly convex w.r.t. a norm  $\|\cdot\|$ , if for all  $x, y \in \mathbb{R}^n$  it holds that

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x) + \frac{\alpha}{2} \|x - y\|^2.$$

**Definition 2.** We say that a function  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $\beta$ -smooth w.r.t. a norm  $\|\cdot\|$ , if for all  $x, y \in \mathbb{R}^n$  it holds that

$$f(y) \leq f(x) + \nabla f(x) \cdot (y - x) + \frac{\beta}{2} \|x - y\|^2.$$

The first-order optimality condition implies that for a  $\alpha$ -strongly convex  $f$ , if  $x^*$  is the unique minimizer of  $f$  over a convex and compact set  $\mathcal{K} \subset \mathbb{R}^n$ , then for all  $x \in \mathcal{K}$  it holds that

$$f(x) - f(x^*) \geq \frac{\alpha}{2} \|x - x^*\|^2. \quad (1)$$

Throughout this work we let  $\|\cdot\|$  denote the standard euclidean norm. Given a point  $x \in \mathbb{R}^n$ , we let  $\text{card}(x)$  denote the number of non-zero entries in  $x$ .

## 2.1 Setting

In this work we consider the following optimization problem:

$$\min_{x \in \mathcal{P}} f(x).$$

We make the following assumptions on  $f$  and  $\mathcal{P}$ :

- $f(x)$  is  $\alpha$ -strongly convex and  $\beta$ -smooth with respect to the  $\ell_2$  norm.
- $\mathcal{P}$  is a polytope which satisfies the following two properties:
  1.  $\mathcal{P}$  can be described algebraically as  $\mathcal{P} = \{x \in \mathbb{R}^n \mid x \geq 0, Ax = b\}$ .
  2. All vertices of  $\mathcal{P}$  lie on the hypercube  $\{0, 1\}^n$ .

We let  $x^*$  denote the (unique) minimizer of  $f$  over  $\mathcal{P}$ , and we let  $D$  denote the Euclidean diameter of  $\mathcal{P}$ , namely,  $D = \max_{x, y \in \mathcal{P}} \|x - y\|$ . We let  $\mathcal{V}$  denote the set of vertices of  $\mathcal{P}$ , where according to our assumptions, it holds that  $\mathcal{V} \subset \{0, 1\}^n$ .

While the polytopes that satisfy the above assumptions are not completely general, these assumptions already capture several important concepts such as paths in graphs, perfect-matchings, Markov random fields, and more. Indeed, a surprisingly large number of applications from machine learning, signal processing and other domains formulate optimization problems in this category (e.g., [12, 14, 16]). We give detailed examples of such polytopes in Section 4. Importantly, the above assumptions allow us to get rid of the dependency of the convergence rate on certain geometric parameters (such as  $\psi, \xi$  in [6] or the *pyramidal width* in [15, 16]), which can be polynomial in the dimension, and hence result in an impractical convergence rate. Finally, for many of these polytopes, the vertices are sparse, i.e., for any vertex  $v \in \mathcal{V}$ ,  $\text{card}(v) \ll n$ . In this case, when the optimum  $x^*$  can be decomposed as a convex combination of only a few vertices (and thus, sparse by itself), we get a sharper convergence rate that depends on the sparsity of  $x^*$  and not explicitly on the dimension, as in previous works.

We believe that our theoretical guarantees could be well extended to more general polytopes and we leave this extension for future work.

## 3 Our Approach

In order to better communicate our ideas, we begin by first briefly introducing the standard conditional gradient method and its accelerated away-steps-based variants. We discuss both the blessings and shortcomings of these away-steps-based variants in Subsection 3.1. Then, in Subsection 3.2, we present our new method, a decomposition-invariant away-steps-based conditional gradient algorithm, and discuss how it addresses the major shortcomings of previous away-steps-based variants.

### 3.1 The conditional gradient method and acceleration via away-steps

The standard conditional gradient algorithm is given below (Algorithm 1). It is well known that when setting the step-size  $\eta_t$  in an appropriate way, the worst case convergence rate of the method is  $O(\beta D^2/t)$  [12]. This convergence rate is tight for the method in general, see for instance [18].

Consider the iterate of Algorithm 1 on iteration  $t$ , and let  $x_t = \sum_{i=1}^k \lambda_i v_i$  be its convex decomposition into vertices of the polytope  $\mathcal{P}$ . Note that Algorithm 1, implicitly discounts each coefficient  $\lambda_i$  by a factor  $(1 - \eta_t)$ , in favor of the new added vertex  $v_t$ . A different approach,

---

**Algorithm 1** Conditional Gradient

---

```
1: Let  $x_1$  be some vertex in  $\mathcal{V}$ 
2: for  $t = 1 \dots$  do
3:    $v_t \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_t)$ 
4:   choose a step-size  $\eta_t \in (0, 1]$ 
5:    $x_{t+1} \leftarrow (1 - \eta_t)x_t + \eta_t v_t$ 
6: end for
```

---

is not to decrease all vertices in the decomposition of  $x_t$  uniformly, but to more-aggressively decrease vertices that are worse than others, with respect to some computable measure, such as their product with the gradient direction. This key principle proves to be crucial to breaking the  $1/t$  rate of the standard method, and to achieve a linear convergence rate under certain strong-convexity assumptions, as described in the recent works [6, 16, 2]. For instance, in [6] it was shown, via the introduction of the concept of a *Local Linear Optimization Oracle*, that using such a non-uniform reweighing rule, in fact approximates a certain *proximal* problem, that together with the shrinking effect of strong convexity, as captured by Eq. (1), yields a linear convergence rate. We refer to these methods as away-step-based CG methods. As a concrete example, which will also serve as a basis for our new method, we bring the *pairwise* variant recently studied in [16], which applies this principle in Algorithm 2, given below <sup>1</sup>. Note that Algorithm 2 decreases the weight of exactly one vertex in the decomposition: that with the largest product with the gradient.

It is important to note that since previous away-step-based CG, unlike the original CG method, do not decrease the coefficients in the convex decomposition of the current iterate uniformly, they all require to explicitly store and maintain a convex decomposition of the current iterate. This issue raises two main disadvantages:

**Superlinear memory and running-time overheads** Storing a decomposition of the current iterate as a convex combination of vertices of the polytope generally requires  $O(n^2)$  memory. While the away-step-based variants increase the size of the decomposition by at most a single vertex per iteration, they also typically exhibit linear convergence after performing at least  $O(n)$  steps [6, 16, 2], and thus, this  $O(n^2)$  estimate still holds. Moreover, since these methods require i) to find the worse vertex in the decomposition, in terms of dot-product with current gradient direction, and ii) to update this decomposition on each iteration (even when using sophisticated update techniques such as in [2]), the per-iteration over-head in terms of computation time of these methods is also at least  $O(n^2)$ .

**Decomposition-specific performance** While the choice of new vertex to be added in Algorithms 1 is independent of a specific representation of the current iterate  $x_t$  as a convex combination of vertices of the polytope, the choice of away-step in Algorithm 2 does depend on the specific decomposition that is maintained by the algorithm. Since the feasible point  $x_t$  may admit several different convex decompositions, committing to one such decomposition, might result in sub-optimal away-steps. Ideally, the away-steps, much like the standard CG methods, will be independent of any specific decomposition. As observable in Table 1, for certain problems in which the optimal solution is sparse, all analyses of previous away-steps-based variants are significantly suboptimal, since they all depend explicitly on the dimension, which seems to

---

<sup>1</sup>While the convergence rate of this pairwise variant, established in [16], despite being linear, is significantly worse than other away-step-based variants, here we show on the contrary, that a proper analysis yields state-of-the-art performance guarantees.

be an unavoidable side-effect of being decomposition-dependent. On the other hand, the fact that our new approach is decomposition-invariant allows us to obtain sharper convergence rates for such instances.

---

**Algorithm 2** Pairwise Conditional Gradient

---

- 1: Let  $x_1$  be some vertex in  $\mathcal{V}$
  - 2: **for**  $t = 1 \dots$  **do**
  - 3:   let  $\sum_{i=1}^{k_t} a_t^{(i)} v_t^{(i)}$  be an **explicitly maintained** convex decomposition of  $x_t$
  - 4:    $v_t^+ \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_t)$
  - 5:    $j_t \leftarrow \arg \min_{j \in [k_t]} v_t^{(j)} \cdot (-\nabla f(x_t))$
  - 6:   choose a step-size  $\eta_t \in (0, a_t^{(j_t)}]$
  - 7:    $x_{t+1} \leftarrow x_t + \eta_t (v_t^+ - v_t^{(j_t)})$
  - 8:   update the convex decomposition of  $x_{t+1}$
  - 9: **end for**
- 

### 3.2 A new decomposition-invariant pairwise conditional gradient method

Our main observation is that in many cases of interest, given a feasible iterate  $x_t$ , one can in-fact compute an optimal away-step from  $x_t$  without relying on any single specific decomposition. This observation allows us to overcome both of the main disadvantages of previous away-step-based CG variants. Our algorithm, which we refer to as a *decomposition-invariant pairwise conditional gradient* (DICG), is given below in Algorithm 3.

---

**Algorithm 3** Decomposition-invariant Pairwise Conditional Gradient

---

- 1: input: sequence of step-sizes  $\{\eta_t\}_{t \geq 1}$
- 2: let  $x_0$  be an arbitrary point in  $\mathcal{P}$
- 3:  $x_1 \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_0)$
- 4: **for**  $t = 1 \dots$  **do**
- 5:    $v_t^+ \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_t)$
- 6:   define the vector  $\tilde{\nabla} f(x_t) \in \mathbb{R}^m$  as follows:

$$\tilde{\nabla} f(x_t)_i := \begin{cases} \nabla f(x_t)_i & \text{if } x_t > 0 \\ -\infty & \text{if } x_t = 0 \end{cases}$$

- 7:    $v_t^- \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot (-\tilde{\nabla} f(x_t))$
- 8:   choose a new step-size  $\tilde{\eta}_t$  using one of the following two options:

**Option 1: predefined step-size**

let  $\delta_t$  be the smallest natural such that  $2^{-\delta_t} \leq \eta_t$ , and set a new step-size  $\tilde{\eta}_t \leftarrow 2^{-\delta_t}$

**Option 2: line-search**

$\gamma_t \leftarrow \max_{\gamma \in [0,1]} \{x_t + \gamma(v_t^+ - v_t^-) \geq 0\}$ ,  $\tilde{\eta}_t \leftarrow \min_{\eta \in (0, \gamma_t]} f(x_t + \eta(v_t^+ - v_t^-))$

- 9:    $x_{t+1} \leftarrow x_t + \tilde{\eta}_t (v_t^+ - v_t^-)$
  - 10: **end for**
- 

The following observation details the optimality of away-steps taken by Algorithm 3.

**Observation 1** (optimal away-steps in Algorithm 3). *Consider an iteration  $t$  of Algorithm 3 and suppose that the iterate  $x_t$  is feasible. Let  $x_t = \sum_{i=1}^k \lambda_i v_i$  for some integer  $k$ , be an*

irreducible way of writing  $x_t$  as a convex sum of vertices of  $\mathcal{P}$ , i.e.,  $\lambda_i > 0$  for all  $i \in [k]$ . Then it holds that

$$\forall i \in [k] : \quad v_i \cdot \nabla f(x_t) \leq v_t^- \cdot \nabla f(x_t), \quad \gamma_t \geq \min\{x_t(i) \mid i \in [n], x_t(i) > 0\}.$$

*Proof.* Let  $x_t = \sum_{i=1}^k \lambda_i v_i$  be a convex decomposition of  $x_t$  into vertices of  $\mathcal{P}$ , for some integer  $k$ , where each  $\lambda_i$  is positive. Note that it must hold that for any  $j \in [n]$  and any  $i \in [k]$ ,  $x_t(j) = 0 \Rightarrow v_i(j) = 0$ , since by our assumption on  $\mathcal{P}$ ,  $\mathcal{V} \subset \mathbb{R}_+^n$ . The observation then follows directly from the definition of  $v_t^-$ .  $\square$

The following theorem which details the convergence rate of Algorithm 3 is the main theorem of this paper.

**Theorem 1.** Let  $M_1 = \sqrt{\frac{\alpha}{8\text{card}(x^*)}}$  and  $M_2 = \frac{\beta D^2}{2}$ . Consider running Algorithm 3 with Option 1 for the step-size, and suppose that

$$\forall t \geq 1 : \quad \eta_t = \frac{M_1}{2\sqrt{M_2}} \left(1 - \frac{M_1^2}{4M_2}\right)^{\frac{t-1}{2}}.$$

Then the iterates of Algorithm 3 are always feasible and satisfy:

$$\forall t \geq 1 : \quad f(x_t) - f(x^*) \leq \frac{\beta D^2}{2} \exp\left(-\frac{\alpha}{8\beta D^2 \text{card}(x^*)} t\right).$$

The following corollary of Theorem 1 shows that the so-called *duality gap*, defined as  $g_t := (x_t - v_t^+) \cdot \nabla f(x_t)$ , which serve as a certificate of the sub-optimality of the iterates of Algorithm 3, also converges with a linear rate.

**Corollary 1.** For any iteration  $t$  of Algorithm 3, define the dual gap  $g_t := (x_t - v_t^+) \cdot \nabla f(x_t)$ , and observe that, since  $f(x)$  is convex,  $h_t \leq g_t$ . Then, for any  $t$  which satisfies:  $h_t \leq \frac{\beta D^2}{2}$ , it holds that

$$g_t \leq \sqrt{2\beta D^2 h_t}.$$

We now turn to make several remarks regarding Algorithm 3 and Theorem 1:

- Note that on any iteration  $t$  of the algorithm, aside from the computation of the gradient vector  $\nabla f(x_t)$  and the two calls to the linear optimization oracle of  $\mathcal{P}$ , all other computations, when using the first option for choosing the step-size, can be carried out in  $O(n)$  time and space. This is much more efficient than previous linearly convergent CG variant, such as those in [6, 16, 2], which typically require at least additional  $O(n^2)$  time and space per iteration, since they require to maintain an explicit convex decomposition of the iterates.
- Note that despite the different parameters of the problem at hand (e.g.,  $\alpha, \beta, D, \text{card}(x^*)$ ), running the algorithm with Option 1 for choosing the step-size, for which the guarantee of Theorem 1 holds, requires the knowledge of a *single* parameter, i.e.,  $M_1/\sqrt{M_2}$ . In particular, it is an easy consequence that running the algorithm with an estimate  $M \in [0.5M_1/\sqrt{M_2}, M_1/\sqrt{M_2}]$ , will only affect the leading constant in the convergence rate listed in the theorem. Hence,  $M_1/\sqrt{M_2}$  could be efficiently estimated via a logarithmic-scale search.
- Theorem 1 improves significantly over the convergence rate established for the pairwise conditional gradient variant in [16]. In particular, the number of iterations to reach an  $\epsilon$  error in the analysis of [16] depends linearly on  $|\mathcal{V}|!$ , where  $|\mathcal{V}|$  is the number of vertices of  $\mathcal{P}$ .



## 4 Examples of Polytopes

In this section we turn to survey several important examples of structured polytopes that fit the assumptions detailed in Subsection 2.1 and detail the application of Algorithm 3 to optimization over these polytopes.

**Unit Simplex** The simplex in  $\mathbb{R}^n$  is the set of all distributions over  $n$  elements, i.e. the set:

$$\mathcal{S}_n = \{x \in \mathbb{R}^n \mid x \geq 0, \sum_{i=1}^n x_i = 1\}.$$

Alternatively,  $\mathcal{S}_n$  is the convex hull of all standard basis vectors in  $\mathbb{R}^n$ .

It is easy to verify that  $D = \sqrt{2}$ .

Linear minimization over the simplex is trivial and can be carried out by a single pass over the non-zero elements in the linear objective. In particular, computing  $v_t^-$  in Algorithm 3 simply amounts to finding the largest (signed) entry in  $\nabla f(x_t)$  which corresponds to a non-zero entry in  $x_t$ , and thus is even more efficient than computing the standard CG direction  $v_t^+$ .

**Flow polytope** Let  $G$  be a *directed acyclic graph* (DAG) with a set of vertices  $V$  such that  $|V| = n$ , and a set of edges  $E$  such that  $|E| = m$ , and let  $s, t$  be two vertices in  $V$  which we refer to as the *source* and the *target*, respectively. The  $s - t$  flow polytope, denoted here by  $\mathcal{F}_{st}$ , is the set of all unit  $s - t$  flows in  $G$ , where for each point  $x \in \mathcal{F}_{st}$  and  $i \in [m]$ , the entry  $x_i$  is the amount of flow through edge  $i$  according to the flow  $x$ .  $\mathcal{F}_{st}$  is also known as the  $s - t$  *path polytope* since it is the convex hull of all identifying vectors of paths from  $s$  to  $t$  in the graph  $G$ . It is easy to verify that that  $D < \sqrt{2n}$ .

Since  $\mathcal{F}_{st}$  is the convex hull of paths, linear minimization is straightforward: given a linear objective  $c \in \mathbb{R}^m$ , we need to find the identifying vector of the lightest  $s - t$  path in  $G$  with respect to the edge weights induced by  $c$ . Since the graph  $G$  is a DAG, this could be carried out in  $O(m)$  time [22]. In particular, computing the direction  $v_t^-$  in Algorithm 3 over the flow polytope, amounts to finding the lightest  $s - t$  path in  $G$  with respect to the gradient vector  $\nabla f(x_t)$ , under the constraint that all edges on the path are assigned non-zero flow by  $x_t$ . Thus, we can compute  $v_t^-$  by running a shortest  $s - t$  path algorithm after removing all edges with zero flow from the graph. Thus, as in the simplex case, computing  $v_t^-$  is even more efficient than computing the standard direction  $v_t^+$ .

It is also important to note that when  $G$  is not extremely sparse, i.e., when  $m = \omega(n)$ , it holds for every vertex  $v$  of  $\mathcal{F}_{st}$  that  $\text{card}(v) \ll m$ . Thus, if  $x^*$  can be expressed as a combination of only a few paths, i.e., it corresponds to a sparse flow, it holds that  $\text{card}(x^*)$  is much smaller than the standard dimension of the problem  $m$ .

**Perfect Matchings polytope** Let  $G$  be a *bipartite* graph with  $n$  vertices on each side and  $m$  crossing edges. The perfect matching polytope, denoted here by  $\mathcal{M}$ , is the convex hull of all identifying vectors of perfect matchings in  $G$ . In case the two sides of  $G$  are fully connected, this polytope is also known as the Birkhoff polytope - the set of all  $n \times n$  doubly stochastic matrices, i.e. matrices with non-negative real entries whose entries along any row and any column add up to 1. It easily follows that  $D \leq \sqrt{2n}$ .

In order to minimize a linear objective over  $\mathcal{M}$ , we need to find a minimum-weight perfect matching in a bipartite graph, where the edge weights are induced by the linear objective. This could be carried out via combinatorial algorithms in  $\min\{\tilde{O}(\sqrt{nm}), O(n^3)\}$  time [22]. As in the flow polytope, in this case also, computing  $v_t^-$  is even more efficient than computing  $v_t^+$ , since it

amounts to finding a minimum weight perfect matching after all edges that are zero-valued in  $x_t$  are removed from the graph.

As in the flow polytope, in case  $G$  is not trivially sparse, i.e., when  $m = \omega(n)$ , it holds that if  $x^*$  could be expressed as a combination of only a few matchings in  $G$ , then  $\text{card}(x^*) \ll m$ , where  $m$  is the dimension of the problem.

**Marginal polytope** In *Graphical Models* several optimization problems are defined for variables representing marginal distributions over subsets of model variables. There exists a set of linear constraints, known as the *marginal polytope*, which guarantees that these variables are legal marginals of some global distribution [25]. For example, the learning problem in Max-Margin Markov Networks is defined as a quadratic program over the marginal polytope [24].

For general graphical models the marginal polytope consists of an exponential number of constraints. Fortunately, for some models, such as tree-structured graphs, the polytope can be characterized by a polynomial number of local consistency constraints, known as the *local marginal polytope* [25]. Consider a set of discrete variables  $(y_1, \dots, y_n)$ , and denote by  $\mu_c(y_c)$  the marginal probability of an assignment to a subset of these variables  $y_c$ . Then the local marginal polytope is defined as:

$$\mathcal{M}_L = \left\{ \mu \geq 0 : \begin{array}{ll} \sum_{y_{c \setminus i}} \mu_c(y_c) = \mu_i(y_i) & \forall c, i \in c, y_i \\ \sum_{y_i} \mu_i(y_i) = 1 & \forall i \end{array} \right\}$$

For tree-structured graphs  $\mathcal{M}_L$  is known to have only integral vertices [25], so it has the desired form assumed in Section 2.1.

In this case  $D = \sqrt{2}|\mathcal{C}|$ , where  $\mathcal{C}$  is the number of subsets  $y_c$  (factors in the graphical model).

In many interesting cases linear optimization over the marginal polytope can be implemented efficiently via dynamic programming. For example, for chain-structured graphs the Viterbi algorithm is used. Finally, we note that computing the direction  $v_t^-$  in Algorithm 3 can often be cheaper than computing  $v_t^+$ , since the restriction to the support of  $x_t$  can eliminate many of the possible configurations of marginals.

## 5 Analysis

In this section we turn to analyze the performance of Algorithm 3, and prove Theorem 1.

Throughout this section we let  $h_t$  denote the approximation error of Algorithm 3 on iteration  $t$ , for any  $t \geq 1$ , i.e.,  $h_t = f(x_t) - f(x^*)$ .

### 5.1 Feasibility of the iterates generated by Algorithm 3

We start by proving that the iterates of Algorithm 3 are always feasible. While feasibility is straightforward when using the the line-search option to set the step-size (Option 2), it is less obvious when using the first option.

**Observation 2.** *Suppose that on some iteration  $t$  of Algorithm 3, the iterate  $x_t$  is feasible, and that the step-size is chosen using Option 1. Then, if for all  $i \in [n]$  for which  $x_t(i) \neq 0$  it holds that  $x_t(i) \geq \tilde{\eta}_t$ , then the following iterate  $x_{t+1}$  is also feasible.*

*Proof.* From the optimality of  $v_t^-$  it follows that for any  $i \in [n]$ , if  $x_t(i) = 0$ , then  $v_t^-(i) = 0$  (note in particular that any vertex with positive weight in some convex decomposition of  $x_t$  must satisfy this condition). Thus, from our assumption on the size of positive entries in  $x_t$ ,

and since  $v_t^- \in \{0, 1\}^n$ , it follows that the vector  $w_t := x_t - \tilde{\eta}_t v_t^-$ , satisfies:  $w_t \geq 0$ . Since  $v_t^+$  is feasible it also follows that  $x_{t+1} = w_t + \tilde{\eta}_t \geq 0$ . Finally, since  $x_t, v_t^-, v_t^+$  are all feasible, it also holds that  $Ax_{t+1} = b$ . Thus,  $x_{t+1}$  is feasible.  $\square$

**Lemma 1** (feasibility of iterates under Option 1). *Suppose that the sequence of step-sizes  $\{\eta_t\}_{t \geq 1}$  is monotonically non-increasing, and contained in the interval  $[0, 1]$ . Then, the iterates generated by Algorithm 3 using Option 1 for setting the step-size, are always feasible.*

*Proof.* We are going to prove by induction that on each iteration  $t$  there exists a non-negative integer-valued vector  $s_t \in \mathbb{N}^n$ , such that for any  $i \in [n]$ , it holds that  $x_t(i) = 2^{-\delta_t} s_t(i)$ . The lemma then follows by applying Observation 2, and since by definition,  $\tilde{\eta}_t = 2^{-\delta_t}$ .

The base case  $t = 1$  holds since  $x_1$  is a vertex of  $\mathcal{P}$  and thus for any  $i \in [n]$  we have that  $x_1(i) \in \{0, 1\}$  (recall that  $\mathcal{V} \subset \{0, 1\}^n$ ). On the other hand, since  $\eta_1 \leq 1$ , it follows that  $\delta_1 \geq 0$ . Thus, there indeed exists a non-negative integer-valued vector  $s_1$ , such that  $x_1 = 2^{-\delta_1} s_1$ .

Suppose now that the induction holds for some  $t \geq 1$ . Since by definition of  $v_t^-$ , subtracting  $\tilde{\eta}_t v_t^-$  from  $x_t$  can only decrease positive entries in  $x_t$  (see proof of Observation 2), and both  $v_t^-, v_t^+$  are vertices of  $\mathcal{P}$  (and thus in  $\{0, 1\}^n$ ), and  $\tilde{\eta}_t = 2^{-\delta_t}$ , it follows that each entry  $i$  in  $x_{t+1}$  is given by:

$$x_{t+1}(i) = 2^{-\delta_t} \begin{cases} s_t(i) & \text{if } s_t(i) \geq 1 \text{ \& } v_t^-(i) = v_t^+(i) = 1 \text{ or } v_t^-(i) = v_t^+(i) = 0 \\ s_t(i) - 1 & \text{if } s_t(i) \geq 1 \text{ \& } v_t^-(i) = 1 \text{ \& } v_t^+(i) = 0 \\ s_t(i) + 1 & \text{if } v_t^-(i) = 0 \text{ \& } v_t^+(i) = 1 \end{cases}$$

Thus,  $x_{t+1}$  can also be written in the form  $2^{-\delta_t} \tilde{s}_{t+1}$  for some  $\tilde{s}_{t+1} \in \mathbb{N}^n$ . By definition of  $\delta_t$  and the monotonicity of  $\{\eta_t\}_{t \geq 1}$ , we have that  $\frac{2^{-\delta_t}}{2^{-\delta_{t+1}}}$  is a positive integer. Thus, setting  $s_{t+1} = \frac{2^{-\delta_t}}{2^{-\delta_{t+1}}} \tilde{s}_{t+1}$ , the induction holds also for  $t + 1$ .  $\square$

## 5.2 Bounding the per-iteration error-reduction of Algorithm 3

The following technical lemma is the key to deriving the linear convergence rate of our method, and in particular, to deriving the improved dependence on the sparsity of  $x^*$ , instead of the dimension. At a high-level, the lemma translates the  $\ell_2$  distance between two feasible points into a  $\ell_1$  distance in a simplex defined over the set of vertices of the polytope, which as we will show, is a natural way to measure distances for conditional gradient-based methods.

**Lemma 2.** *Let  $x, y \in \mathcal{P}$ . There exists a way to write  $x$  as a convex combination of vertices of  $\mathcal{P}$ ,  $x = \sum_{i=1}^k \lambda_i v_i$  for some integer  $k$ , such that  $y$  can be written as  $y = \sum_{i=1}^k (\lambda_i - \Delta_i) v_i + (\sum_{i=1}^k \Delta_i) z$  with  $\Delta_i \in [0, \lambda_i] \forall i \in [k], z \in \mathcal{P}$ , and  $\sum_{i=1}^k \Delta_i \leq \sqrt{\text{card}(y)} \|x - y\|$ .*

*Proof.* Consider writing  $y$  as some convex combination of vertices,  $y = \sum_{i=1}^s \gamma_i u_i$  for some appropriate integer  $s$ . Applying Lemma 5.3. from [6], it follows that we can write  $x$  as

$$x = \sum_{i=1}^s (\gamma_i - \tilde{\Delta}_i) u_i + \left( \sum_{i=1}^s \tilde{\Delta}_i \right) \tilde{z}, \quad (2)$$

where  $\tilde{\Delta}_i \in [0, \gamma_i] \forall i \in [s]$ ,  $\tilde{z} \in \mathcal{P}$ , and for every  $i$  with  $\tilde{\Delta}_i > 0$  there exists  $j_i \in [n]$  such that  $\tilde{z}(j_i) = 0$  and  $u_i(j_i) > 0$ . Since each  $u_i$  is a vertex of  $\mathcal{P}$  and thus a point of the  $\{0, 1\}$ -hypercube, it further follows that  $u_i(j_i) = 1$ . Let  $C = \{j_i \mid i \in [s]\}$ . Observe that  $|C| \leq \text{card}(y)$ . Now, we

have that

$$\begin{aligned}\|x - y\|^2 &= \left\| \sum_{i=1}^s \tilde{\Delta}_i (u_i - \tilde{z}) \right\|^2 \geq \sum_{j \in C} \left( \sum_{i=1}^s \tilde{\Delta}_i (u_i(j) - \tilde{z}(j)) \right)^2 = \sum_{j \in C} \left( \sum_{i=1}^s \tilde{\Delta}_i u_i(j) \right)^2 \\ &\geq \frac{1}{|C|} \left( \sum_{j \in C} \sum_{i=1}^s \tilde{\Delta}_i u_i(j) \right)^2 \geq \frac{1}{|C|} \left( \sum_{i=1}^s \tilde{\Delta}_i \right)^2.\end{aligned}$$

Rearranging we have that

$$\sum_{i=1}^s \tilde{\Delta}_i \leq \sqrt{|C|} \|x - y\| \leq \sqrt{\text{card}(y)} \|x - y\|. \quad (3)$$

Note that using the convex decomposition of  $x$  as in Eq. (2), and the bound in Eq. (3) it follows that we can rewrite  $y$  as a convex decomposition as suggested in the lemma.  $\square$

**Lemma 3.** *Consider the iterates of Algorithm 3, when the step-sizes are chosen using Option 1. Let  $M_1 = \sqrt{\frac{\alpha}{8\text{card}(x^*)}}$  and  $M_2 = \frac{\beta D^2}{2}$ . For any  $t \geq 1$  it holds that*

$$h_{t+1} \leq h_t - \eta_t M_1 h_t^{1/2} + \eta_t^2 M_2.$$

*Proof.* Define  $\Delta_t = \sqrt{\frac{2\text{card}(x^*)h_t}{\alpha}}$ , and note that from Eq. (1) we have that  $\Delta_t \geq \sqrt{\text{card}(x^*)} \|x_t - x^*\|$ .

As a first step, we are going to show that the point  $y_t := x_t + \Delta_t(v_t^+ - v_t^-)$  satisfies:  $y_t \cdot \nabla f(x_t) \leq x^* \cdot \nabla f(x_t)$ .

From Lemma 2 it follows that we can write  $x$  as a convex combination  $x_t = \sum_{i=1}^k \lambda_i v_i$  and write  $x^*$  as  $x^* = \sum_{i=1}^k (\lambda_i - \Delta_i) v_i + \sum_{i=1}^k \Delta_i z$ , where  $\Delta_i \in [0, \lambda_i]$ ,  $z \in \mathcal{P}$ , and  $\sum_{i=1}^k \Delta_i \leq \Delta_t$ . It holds that

$$\begin{aligned}(y_t - x_t) \cdot \nabla f(x_t) &= \Delta_t (v_t^+ - v_t^-) \cdot \nabla f(x_t) \leq \sum_{i=1}^k \Delta_i (v_t^+ - v_t^-) \cdot \nabla f(x_t) \\ &\leq \sum_{i=1}^k \Delta_i (z - v_i) \cdot \nabla f(x_t) = (x^* - x_t) \cdot \nabla f(x_t),\end{aligned}$$

where the first inequality follows since  $(v_t^+ - v_t^-) \cdot \nabla f(x_t) \leq 0$ , and the second inequality follows from the optimality of  $v_t^+$  and  $v_t^-$  (Observation 1). Rearranging, we have that indeed

$$(x_t + \Delta_t(v_t^+ - v_t^-)) \cdot \nabla f(x_t) \leq x^* \cdot \nabla f(x_t), \quad (4)$$

as needed.

Observe now that from the definition of  $\tilde{\eta}_t$  it follows for any  $t \geq 1$  that  $\frac{\eta_t}{2} \leq \tilde{\eta}_t \leq \eta_t$ . Using

the smoothness of  $f(x)$  we have that

$$\begin{aligned}
h_{t+1} &= f(x_t + \tilde{\eta}_t(v_t^+ - v_t^-)) - f(x^*) \\
&\leq h_t + \tilde{\eta}_t(v_t^+ - v_t^-) \cdot \nabla f(x_t) + \frac{\tilde{\eta}_t^2 \beta}{2} \|v_t^+ - v_t^-\|^2 \\
&\leq h_t + \tilde{\eta}_t(v_t^+ - v_t^-) \cdot \nabla f(x_t) + \frac{\tilde{\eta}_t^2 \beta D^2}{2} \\
&\leq h_t + \frac{\eta_t}{2} (v_t^+ - v_t^-) \cdot \nabla f(x_t) + \frac{\eta_t^2 \beta D^2}{2} \\
&= h_t + \frac{\eta_t}{2\Delta_t} ((x_t + \Delta_t(v_t^+ - v_t^-)) - x_t) \cdot \nabla f(x_t) + \frac{\tilde{\eta}_t^2 \beta D^2}{2} \\
&\leq h_t + \frac{\eta_t}{2\Delta_t} (x^* - x_t) \cdot \nabla f(x_t) + \frac{\eta_t^2 \beta D^2}{2} \\
&\leq h_t - \frac{\eta_t}{2\Delta_t} h_t + \frac{\eta_t^2 \beta D^2}{2} \\
&= h_t - \frac{\eta_t \sqrt{\alpha}}{2\sqrt{2\text{card}(x^*)} h_t} h_t + \frac{\eta_t^2 \beta D^2}{2},
\end{aligned}$$

where the third inequality follows since  $(v_t^+ - v_t^-) \cdot \nabla f(x_t) \leq 0$ , the forth inequality follows from Eq. (4), the fifth inequality follows from convexity of  $f(x)$ , and the last equality follows from plugging the value of  $\Delta_t$ .  $\square$

### 5.3 Proof of Theorem 1

We now turn to prove Theorem 1. Afterwards, we prove Corollary 1.

*Proof.* We are first going to prove the convergence rate stated in the theorem, assuming that all iterates are feasible. Then we will show that for our choice of step-sizes, indeed the iterates are feasible. We are going to prove by induction that there exist  $c_0, c_1$  such that for all  $t \geq 1$  it holds that  $h_t \leq c_0(1 - c_1)^{t-1}$ . Clearly for the base case we must require that  $c_0 \geq h_1$ .

Suppose now that the induction holds for some  $t \geq 1$ . Let us set

$$\eta_t = \frac{M_1}{2M_2} \sqrt{c_0(1 - c_1)}^{\frac{t-1}{2}}. \quad (5)$$

Using Lemma 3 and the induction hypothesis we have that

$$\begin{aligned}
h_{t+1} &\leq h_t - \frac{M_1^2}{2M_2} \sqrt{c_0(1 - c_1)^{t-1}} h_t^{1/2} + \frac{M_1^2}{4M_2} c_0(1 - c_1)^{t-1} \\
&\leq h_t - \frac{M_1^2}{2M_2} h_t + \frac{M_1^2}{4M_2} c_0(1 - c_1)^{t-1} \\
&= h_t \left(1 - \frac{M_1^2}{2M_2}\right) + \frac{M_1^2}{4M_2} c_0(1 - c_1)^{t-1} \\
&\leq c_0(1 - c_1)^{t-1} \left(1 - \frac{M_1^2}{4M_2}\right),
\end{aligned}$$

where the induction hypothesis was used in both the second and third inequalities. In the third inequality we have also used the fact that

$$\frac{M_1^2}{2M_2} = \frac{\alpha}{8\beta\text{card}(x^*)D^2} < 1, \quad (6)$$

where the inequality follows since  $\alpha \leq \beta$  and both  $\text{card}(x^*)$ ,  $D$  are at least 1.

Thus, if we set  $c_1 = \frac{M_1^2}{4M_2}$ , the induction follows.

We now turn to figure out  $c_0$ .

Using the smoothness of  $f(x)$  and the choice of  $x_1$  in Algorithm 3, we have that

$$\begin{aligned} h_1 &= f(x_1) - f(x^*) = f(x_0 + (x_1 - x_0)) - f(x^*) \\ &\leq f(x_0) - f(x^*) + (x_1 - x_0) \cdot \nabla f(x_0) + \frac{\beta \|x_0 - x_1\|^2}{2} \\ &\leq f(x_0) - f(x^*) + (x^* - x_0) \cdot \nabla f(x_0) + \frac{\beta \|x_0 - x_1\|^2}{2} \leq \frac{\beta D^2}{2}, \end{aligned}$$

where the last inequality follows from the convexity of  $f(x)$ .

Thus, we can set  $c_0 = \frac{\beta D^2}{2} = M_2$ , which completes the proof of the convergence rate.

Now, it remains to prove that indeed all iterates are feasible. First note that the sequence  $\{\eta_t\}_{t \geq 1}$ , as defined in Eq. (5) is monotonically non-increasing. Furthermore, plugging the values  $M_1, M_2, c_0$ , we have that

$$\eta_1 = \frac{M_1 \sqrt{c_0}}{2M_2} = \frac{1}{2} \sqrt{\frac{M_1^2}{M_2}} = \frac{1}{2} \sqrt{\frac{\alpha}{4\beta D^2 \text{card}(x^*)}} \leq 1,$$

where the inequality follows similarly to the one in Eq. (6). Thus, our choice of step-size sequence  $\{\eta_t\}_{t \geq 1}$  satisfies the conditions of Lemma 1, and thus it follows that all iterates of Algorithm 3 are feasible.  $\square$

We now prove Corollary 1.

*Proof.* Fix an iteration  $t$ . Using the  $\beta$ -smoothness of  $f(x)$  we have that

$$\forall \eta \in (0, 1] : \quad f(x^*) \leq f(x_t + \eta(v_t^+ - x_t)) \leq f(x_t) + \eta(v_t^+ - x_t) \cdot \nabla f(x_t) + \frac{\eta^2 \beta D^2}{2}.$$

Rearranging we have that

$$\forall \eta \in (0, 1] : \quad g_t = (x_t - v_t^+) \cdot \nabla f(x_t) \leq \frac{1}{\eta} h_t + \frac{\eta \beta D^2}{2}.$$

Thus, when  $\sqrt{\frac{2h_t}{\beta D^2}} \leq 1$ , we can set  $\eta = \sqrt{\frac{2h_t}{\beta D^2}}$  in the above inequality, and obtain the corollary.  $\square$

## 6 Extensions

In this section we detail two extensions of our result: i) relaxing the specific structure of the polytope  $\mathcal{P}$  considered in Subsection 2.1, and ii) relaxing the strong convexity requirement on the objective function  $f(x)$ .

### 6.1 Extension of Algorithm 3 to arbitrary polytopes

In this subsection we detail how to extend our approach to a broader class of polytopes. While proving rigorous guarantees for this extension is beyond the scope of this paper and left for future work, the encouraging experimental results for Algorithm 3 with line-search, suggest that this extended variant, for which line-search is also possible, may also exhibit favorable

empirical performance. Towards this end, in this subsection we consider minimizing a smooth and strongly-convex function over an arbitrary polytope  $\mathcal{P}$  which we assume is given in the following way:

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid A_1 x = b_1, A_2 x \leq b_2\},$$

where  $A_2$  is  $m \times n$ . We assume that given a point  $x \in \mathbb{R}^n$ , we have an efficient way to evaluate the vector  $A_2 x$ , which is indeed the case for most structured polytopes of interest.

---

**Algorithm 4** Decomposition-invariant Pairwise Conditional Gradient with Line-search for Arbitrary Polytopes

---

- 1: let  $x_0$  be an arbitrary point in  $\mathcal{P}$
- 2:  $x_1 \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_0)$
- 3: **for**  $t = 1 \dots$  **do**
- 4:    $v_t^+ \leftarrow \arg \min_{v \in \mathcal{V}} v \cdot \nabla f(x_t)$
- 5:   define the vector  $c \in \mathbb{R}^m$  as follows:

$$c_i := \begin{cases} 0 & \text{if } A_2(i) \cdot x_t < b_2(i) \\ \infty & \text{if } A_2(i) \cdot x_t = b_2(i) \end{cases}$$

- 6:    $v_t^- \leftarrow \arg \min_{v \in \mathcal{V}} (-\nabla f(x_t)) \cdot v + c^\top A_2 v$
  - 7:    $\gamma_t \leftarrow \max\{\gamma \in [0, 1] \mid A_2(x_t + \gamma(v_t^+ - v_t^-)) \leq b_2\}$
  - 8:    $\eta_t \leftarrow \arg \min_{\eta \in [0, \gamma_t]} f(x_t + \eta(v_t^+ - v_t^-))$
  - 9:    $x_{t+1} \leftarrow x_t + \eta_t(v_t^+ - v_t^-)$
  - 10: **end for**
- 

**Observation 3** (optimal away-step for an arbitrary polytope). *Consider an iteration  $t$  of Algorithm 4 and suppose that the iterate  $x_t$  is feasible. Let  $x_t = \sum_{i=1}^k \lambda_i v_i$  for some integer  $k$ , be a irreducible way of writing  $x_t$  as a convex sum of vertices of  $\mathcal{P}$ , i.e.,  $\lambda_i > 0$  for all  $i \in [k]$ . Then it holds that*

$$\forall i \in [k]: \quad v_i \cdot \nabla f(x_t) \leq v_t^- \cdot \nabla f(x_t), \quad \gamma_t > 0.$$

Moreover, there exists a convex decomposition of  $x_t$  that assigns a weight at least  $\gamma_t$  to  $v_t^-$ .

*Proof.* Let  $x_t = \sum_{i=1}^k \lambda_i v_i$  be a decomposition of  $x_t$  into vertices of  $\mathcal{P}$  such that  $\lambda_i > 0$  for all  $i \in [k]$ . Observe that for any  $j \in [m]$  and  $i \in [k]$  it holds that  $A_2(j) \cdot x_t = b_2(j) \Rightarrow A_2(j) \cdot v_i = b_2(j)$ . Note that by definition of the vector  $c$  and  $v_t^-$  it holds that

$$\begin{aligned} v_t^- &\in \arg \max_{v \in \mathcal{V}} \nabla f(x_t) \cdot v - c^\top A_2 v \equiv \arg \max_{v \in \mathcal{V}} \nabla f(x_t) \cdot v + c^\top (b_2 - A_2 v) \\ &\equiv \arg \max_{v \in \{y \in \mathcal{V} \mid \forall j \in [m]: A_2(j) \cdot x_t = b_2(j) \Rightarrow A_2(j) \cdot y = b_2(j)\}} v \cdot \nabla f(x_t). \end{aligned} \tag{7}$$

Thus, it follows that for all  $i \in [k]$ ,  $v_t^- \cdot \nabla f(x_t) \geq v_i \cdot \nabla f(x_t)$ .

In order to prove the second part of the observation, we note that from the RHS of Eq. (7) it follows that there exists  $\gamma_t > 0$  such that indeed  $x_t - \gamma_t v_t^- \leq (1 - \gamma_t)b_2$ . To see this, consider some  $j \in [m]$ . If  $A_2(j) \cdot x_t = b_2(j)$ , then from the RHS of Eq. (7), it follows that  $A_2(j) \cdot v_t^- = b_2(j)$  and thus, for any  $\gamma_t$  it holds that  $A_2(j) \cdot (x_t - \gamma_t v_t^-) = (1 - \gamma_t)b_2(j)$ . Otherwise, there exists some  $\epsilon_j > 0$  such that  $A_2(j) \cdot v_t^- \leq b_2(j) - \epsilon_j$ . Thus, for small enough, yet positive  $\gamma_t$  we will have that  $A_2(j) \cdot (x_t - \gamma_t v_t^-) \leq (1 - \gamma_t)b_2(j)$ . Since it clearly also holds that  $A_1(x_t - \gamma_t v_t^-) = (1 - \gamma_t)b_1$ , we have that the vector  $w_t := x_t - \gamma_t v_t^-$  satisfies:  $w_t \in (1 - \gamma_t)\mathcal{P}$ . Hence,  $w_t$  can be decomposed

as  $w_t = \sum_{i=1}^q \tilde{\gamma}_i \tilde{v}_i$ , where  $q$  is a positive integer and for all  $i \in [q]$ ,  $\tilde{\lambda}_i > 0$ ,  $\tilde{v}_i$  is a vertex of  $\mathcal{P}$ , and  $\sum_{i=1}^q \tilde{\lambda}_i = 1 - \gamma_t$ . Thus, since  $v_t^-$  is a vertex of  $\mathcal{P}$ , it follows that  $x_t = w_t + \gamma_t v_t^-$  admits the convex decomposition  $\sum_{i=1}^q \tilde{\lambda}_i \tilde{v}_i + \gamma_t v_t^-$ , as needed.  $\square$

The following lemma is an immediate consequence of the choice of  $\gamma_t$  in Algorithm 4.

**Lemma 4.** *The iterates of Algorithm 4 are always feasible.*

## 6.2 Relaxing the strong convexity of the objective function

Until now we have assumed that the objective function  $f$  is strongly convex. However, as can be observed from our analysis, the only consequence of strong convexity that we relied on in our analysis, is Eq. (1). Indeed, there exist functions which are not strongly convex, that under certain conditions, still satisfy Eq. (1), and thus are compatible with our method and analysis.

Following the work of Beck and Shtern [2], we can consider a broader class of objective functions, namely functions that take the following form:

$$f(x) = g(Ax) + b \cdot x, \quad (8)$$

where  $A \in \mathbb{R}^{m \times n}$ , and  $g : \mathbb{R}^m \rightarrow \mathbb{R}$  is smooth and strongly convex.

In [2] (Lemma 2.5) it was shown, using an application of Hoffman's lemma, that there exists a constant  $\kappa$  which depends both on the condition number of  $g$  and the parameters  $A, b$ , such that for any feasible point  $x$ , it holds that

$$\min_{y \in \mathcal{P}^*} \|x - y\|^2 \leq \kappa (f(x) - f^*), \quad (9)$$

where  $\mathcal{P}^* \subset \mathcal{P}$ , is the set of all feasible points that minimize  $f(x)$  over  $\mathcal{P}$ , and  $f^*$  is the minimum value of  $f(x)$  over  $\mathcal{P}$ .

It is easy to verify that Eq. (9) can be readily used in our analysis instead of Eq. (1), and thus our results extend to handle objectives of the form given in Eq. (8).

We note that now, the dependency in our analysis and in Theorem 1 on the strong convexity parameter  $\alpha$  will be replaced with  $\kappa$ , and the dependency on  $\text{card}(x^*)$  will be replaced with  $\max_{y \in \mathcal{P}^*} \text{card}(y)$ .

## 7 Lower Bound for Problems with a Sparse Solution

In this section we present a simple lower bound on the approximation error of, informally speaking, any natural conditional gradient variant that when initialized with a vertex of the feasible set, its iterate after  $t$  iterations admits a convex decomposition into at most  $t + 1$  vertices of the polytope. That is, on each iteration, at most a single new vertex is added to the decomposition. The lower bound shows that there exists a 1-smooth and 1-strongly convex function  $f$ , for which, any such CG variant which is applied to the minimization of  $f$  over the unit simplex, must take  $\Omega(\text{card}(x^*))$  steps before entering the linear convergence regime. To date, none of the previous analyses of linearly converging CG variants matches this lower bound since, in this exact setting, they all require, in worst-case,  $\Omega(n)$  steps before entering the linear convergence regime, i.e., number of steps that is independent of  $\text{card}(x^*)$ .

To the best of our knowledge, Algorithm 3 and the corresponding Theorem 1 are the first to match this lower bound. We emphasize that the idea behind the construction of this lower bound is well known and follows almost immediately from previous constructions, such as those in [12, 6].



**Lemma 5.** Fix an even integer  $k \in [n]$ , and consider the optimization problem

$$\min_{x \in \mathcal{S}_n} \{f(x) := \frac{1}{2} \|x - \frac{1}{k} \mathbf{1}_k\|^2\},$$

where  $\mathcal{S}_n$  denotes the unit simplex in  $\mathbb{R}^n$ , i.e.,  $\mathcal{S}_n := \{x \in \mathbb{R}^n \mid x \geq 0, \|x\|_1 = 1\}$ , and  $\mathbf{1}_k$  is a vector in  $\mathbb{R}^n$ , defined as:

$$\mathbf{1}_k(i) = \begin{cases} 1 & \text{if } 1 \leq i \leq k \\ 0 & \text{else} \end{cases}$$

Observe that  $x^* = \frac{1}{k} \mathbf{1}_k$  is the unique minimizer of  $f$  over  $\mathcal{S}_n$ . Then, any point  $x \in \mathcal{S}_n$ , for which it holds that  $\text{card}(x) \leq k/2$  satisfies:

$$f(x) - f(x^*) \geq \frac{1}{4k}.$$

*Proof.* Fix a point  $x \in \mathcal{S}_n$  for which it holds that  $\text{card}(x) \leq k/2$ . In order to lower bound the approximation error of  $x$ , it suffices to consider the entries which are zero for  $x$  and non-zero for  $x^*$ . Thus, we have that

$$f(x) \geq \frac{1}{2} \cdot \frac{k}{2} \cdot \left(0 - \frac{1}{k}\right)^2 = \frac{1}{4k}.$$

□

## 8 Experiments

In this section we illustrate the performance of our algorithm in numerical experiments. We use the two experimental settings from [16], which include a constrained Lasso problem and a video co-localization problem. In addition, we test our algorithm on a learning problem related to an optical character recognition (OCR) task from [24]. In each setting we compare the performance of our algorithm (DICG) to standard conditional gradient (CG), as well as to the fast away (ACG) and pairwise (PCG) variants [16]. For the baselines in the first two settings we use the publicly available code from [16], to which we add our own implementation of Algorithm 3. Similarly, for the OCR problem we extend code from [21], kindly provided by the authors. For all algorithms we use line-search to set the step size.

**Lasso** In the first example the goal is to solve the problem:  $\min_{x \in \mathcal{M}} \|\bar{A}x - \bar{b}\|^2$ , where  $\mathcal{M}$  is a scaled  $\ell_1$  ball. Notice that the constraints  $\mathcal{M}$  do not match the required structure of  $\mathcal{P}$ , however, with a simple change of variables we can obtain an equivalent optimization problem over the simplex. We generate the random matrix  $\bar{A}$  and vector  $\bar{b}$  as in [16]. In Figure 1 (left, top) we observe that our algorithm (DICG) converges similarly to the pairwise variant PCG and faster than the other baselines. This is expected since the away direction  $v^-$  in DICG (Algorithm 3) is equivalent to the away direction in PCG (Algorithm 2) in the case of simplex constraints.

**Video co-localization** The second example is a quadratic program over the flow polytope, originally proposed in [14]. This is an instance of  $\mathcal{P}$  that is mentioned in Section 4 in the appendix. As can be seen in Figure 1 (middle, top), in this setting our proposed algorithm significantly outperforms the baselines, as a result of finding a better away direction  $v^-$ . Figure 1

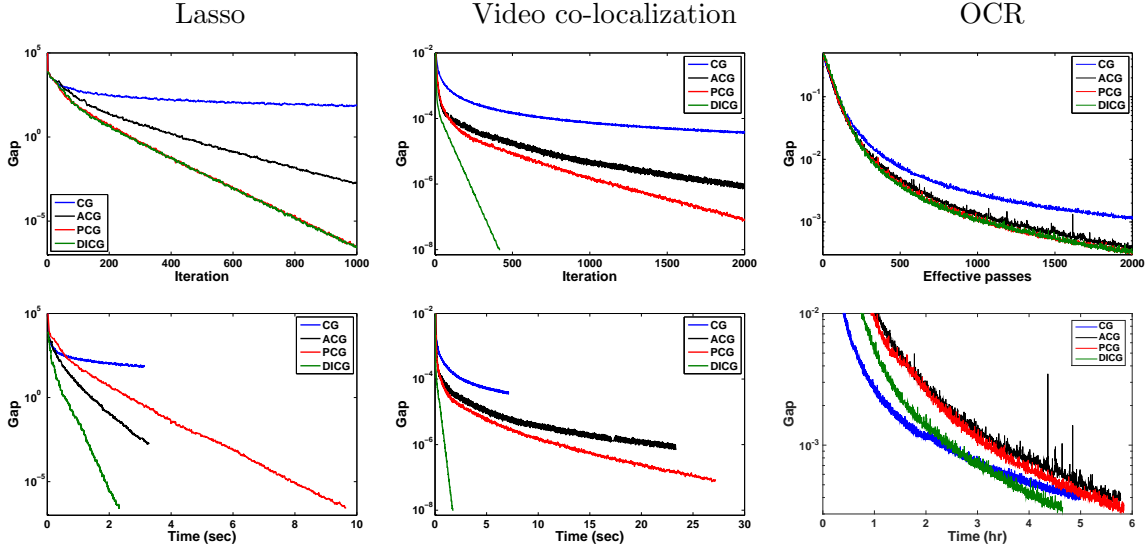


Figure 1: Duality gap  $g_t$  vs. iterations (top) and time (bottom) in various settings.

(middle, bottom) shows convergence on a time scale, where the difference between the algorithms is even larger. One reason for this difference is the costly search over the history of vertices maintained by the baseline algorithms. Specifically, the number of stored vertices grows fast with the number of iterations and reaches 1222 for away steps and 1438 for pairwise steps (out of 2000 iterations).

**OCR** We next conduct experiments on a structured SVM learning problem resulting from an OCR task. The constraints in this setting are the marginal polytope corresponding to a chain graph over the letters of a word (see [24]), and the objective function is quadratic. Notice that the marginal polytope has a concise characterization in this case and also satisfies our assumptions (see Section 4 in the appendix for more details). For this problem we actually run Algorithm 3 in a block-coordinate fashion, where blocks correspond to training examples in the dual SVM formulation [17, 21]. In Figure 1 (right, top) we see that our DICG algorithm is comparable to the PCG algorithm and faster than the other baselines on the iteration scale. Figure 1 (right, bottom) demonstrates that in terms of actual running time we get a noticeable speedup compared to all baselines. We point out that for this OCR problem, both ACG and PCG each require about 5GB of memory to store the explicit decomposition in the implementation that we used, so using DICG instead results in significant memory savings.

## References

- [1] S. Damla Ahipasaoglu, Peng Sun, and Michael J. Todd. Linear convergence of a modified frank-wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software*, 23(1):5–19, 2008.
- [2] Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *arXiv preprint arXiv:1504.05002*, 2015.
- [3] Amir Beck and Marc Teboulle. A conditional gradient method with linear rate of convergence for solving convex linear systems. *Math. Meth. of OR*, 59(2):235–247, 2004.

- [4] Miroslav Dudík, Zaïd Harchaoui, and Jérôme Malick. Lifted coordinate descent for learning with trace-norm regularization. *Journal of Machine Learning Research - Proceedings Track*, 22:327–336, 2012.
- [5] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:149–154, 1956.
- [6] Dan Garber and Elad Hazan. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *CoRR*, abs/1301.4666, 2013.
- [7] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 541–549, 2015.
- [8] Jacques GuéLat and Patrice Marcotte. Some comments on Wolfe’s ‘away step’. *Mathematical Programming*, 35(1), 1986.
- [9] Zaïd Harchaoui, Matthijs Douze, Mattis Paulin, Miroslav Dudík, and Jérôme Malick. Large-scale image classification with trace-norm regularization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2012.
- [10] Elad Hazan and Satyen Kale. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.
- [11] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. *CoRR*, abs/1602.02101, 2016.
- [12] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML*, 2013.
- [13] Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning, ICML*, 2010.
- [14] Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with Frank-Wolfe algorithm. In *Computer Vision—ECCV 2014*, pages 253–268. Springer, 2014.
- [15] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for frank-wolfe algorithms. *CoRR*, abs/1312.7864, 2013.
- [16] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.
- [17] Simon Lacoste-Julien, Martin Jaggi, Mark W. Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *Proceedings of the 30th International Conference on Machine Learning, ICML*, 2013.
- [18] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *CoRR*, abs/1309.5550, 2013.
- [19] Sören Laue. A hybrid algorithm for convex semidefinite optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

- [20] Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6:1–50, 1966.
- [21] Anton Osokin, Jean-Baptiste Alayrac, Puneet K. Dokania, and Simon Lacoste-Julien. Minding the gaps for block frank-wolfe optimization of structured svm. In *International Conference on Machine Learning (ICML)*, 2016.
- [22] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.
- [23] Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. Large-scale convex minimization with a low-rank constraint. In *Proceedings of the 28th International Conference on Machine Learning, ICML*, 2011.
- [24] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.
- [25] M. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.
- [26] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *J. Mach. Learn. Res.*, 13(1):1–26, January 2012.